4 CHOP 元件

4.1 介绍

数据通道类元件，Channel Operators，缩写为 CHOP。是一个负责各种通道类数据处理的元件家族。它可以处理诸如手势交互、音频输入、动画关键帧、硬件输入（Kinect 体感相机、LeapMotion 手势识别设备、Oculus 虚拟现实设备、Pen Table、键盘、麦克、等等）、DMX 调光、MIDI 电子音乐以及 OSC。

这些元件处理输入、运算、输出，并将数据与各种视听设备连接。

例如：

Mixers 调音器、MIDI 控制器、合成器、DMX 调光控制器、Kinect、运行有 TouchDesigner 的其他电脑、扬声器、以及其音视频应用如 Max/MSP，Ableton Live，Resolume。

4.2 通讯方式

MIDI 与现有的很多软硬件有不错的兼用性。数字音乐工作站-DAW,例如 Ableton Live, Avid Pro Tools, Propellerhead Reason 等等，都支持 MIDI 的输入输出。它是一种快速稳定并经过实践考验的协议。调音器通常配备 MIDI USB,它的输入包括按钮、faders、琴键、触摸板、滚轮、节拍器、电位计。

程序运算系统诸如 Cycling 74 Max/MSP, PureData, Native Instruments Reaktor 等等，都已经支持 OSC 通讯。得益于现代网络通讯技术，OSC 有着比 MIDI 更高的性能和更好的架构。OSC 消息可以在 UDP 和 TCP 连接中传输，因此它很容易实现实时长距离的网络传输。目前，OSC 是软件和计算机系统之间通讯最常用的方式。

DMX 是供灯光和控制器之间使用的协议。许多 DMX 设备都有多路调光、灯光跟随、RGB 通道、自动化电机等等。许多灯光控制器用 DMX 与其他设备或电脑通讯。当你打算用这些控制器和平台组建项目时，记得认真参考他们的手册。一般来说，就算有些功能这个项目用不上，你也应当有所了解。有多种方式优化 DMX 的数据收发流程，这些将在后面提到。

同步输入类 CHOP 和同步输出类 CHOP,通常用于 TD 的内外接口的帧同步。他们用 OSC 做底层的通信协议。这两类 OP 通过在每一帧通讯同步状态来工作。当所有同步设备确认，自己已经完成本帧的渲染，他们会同时进入下一帧。通过在每一帧重复这些事件来保证设备间的画面同步。

4.3 音频输入与输出

音频有多种来源和多种处理方式。TouchDesigner 能够使用音频文件，视频文件，外部音频接口，网络音频流等资源，来获取并处理音频。甚至也可以无需来源，自己生成音频。

大多涉及声音设计、音频音轨的项目都会有专用的音频文件。TD 能够读取和播放很多标准的音频文件格式，如 MP3,WAV,AIFF。通过 CHOP 元件中的 Audio File In 和 Audio File Out,这些音频文件可以被循环、修建、重建、插入，以供各种设备使用。

Audio Movie 型 CHOP 可以从视频文件中播放音频文件。不同于从文件中播放音频，该元件引用了一个 Movie In 型 TOP 元件。这样可以使音频与视频保持同步，并有一些参数来使音频更好的匹配视频。

TD 兼容多种外部音频接口，详细的兼容列表，最好去参考 TD 的维基百科和论坛。

这些外部设备可以提供各种模拟和数字音频的输入和输出。人声、乐器、摄像机、调音台、计算机，都可以作为音频的输入源。TD 中与外部音频接口通讯的是 CHOP 元件中的 Audio Device In 以及 Audio Device Out. 他们分别处理输入和输出。 另外还有个 Audio SDI 元件（SDI 指的是 Serial digital interface，此元件只有在购买 TouchDesigner Pro 版本之后才会出现），专门跟英伟达的 Quadro SDI 通讯，采集外部音频。

TD 能访问两种音频驱动。DirectSound 是基于 DirectX 的成熟驱动，被大量使用。而在 TD 088 版后，加入了 ASIO ，它可以改善 DirectX 的一个主要缺陷-必须基于 windows。它能绕过系统，直接与音频设备通讯，从而获得更低的延时。

当在 TD 中同时架起音频输入元件和音频输出元件这两门大炮，传输音频对你来说就像传输数据一样简单了。

4.4 采样率-Sample Rates

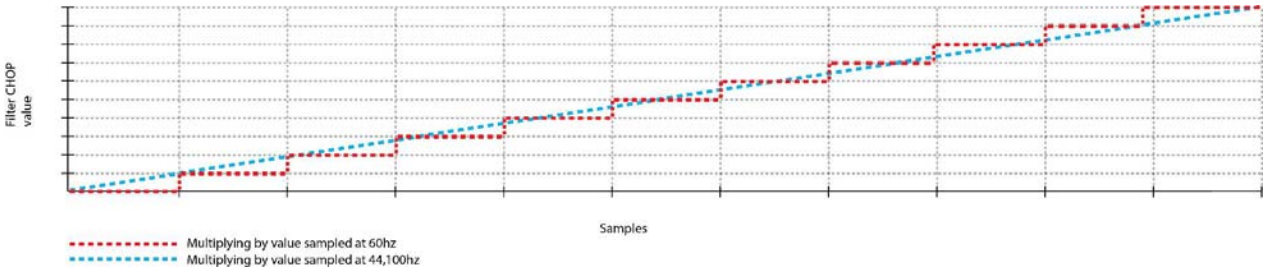
大部分程序不会体现出场景背后的函数与运算，所以大多数并不习惯用数字的方式处理音频。 然而音频的本质，就是被高速运算的数据。认识到这一点，是能够用 TD 处理音频的基础。

打开'Sample_rates_1.toe'，这个例子是音频应用中的一个基本功能:静音。它用 button 的输出值(1 或 0)，和音频流数据相乘来控制静音。

咱们再增加点难度，让音频开关加上淡入淡出效果。 打开'Sample_rates_2.toe'。

这个例子在上一个的基础上，增加了两个元件。一个是 Filter CHOP，它可以在按钮的两种状态之间创建一个平滑处理后的数值。另一个元件是 Resample CHOP。

初学者常常会忽略不同原件之间的采样率的区别。但想要创建可自由编辑音频，必须了解这一点。比如 Oscillator CHOP 的采样率是 44100 次每秒，Filter CHOP 采样率是 60 次每秒。这就意味着，当这两者联用的时候，不可能是 1：1 的采样比，在进行乘法运算的时候，并不会变成一个平稳上升的直线。更准确的说，前者的数据量是后者的 735 倍。也就是说，Oscillator 每过 735 个音频单元，Filter 才会有一次声音上的变化。就像下图中，蓝线是 1：1 的采样比，红线是 735：1 的采样比。



再看上面这张图，当这两个不同采样率的通道相乘，有一个很明显的阶梯变化。很多 CHOP 元件用 FPS 帧速度作为他们的采样单位。到项目的帧速度被设置成 30FPS 的时候，阶梯将会变得更夸张，比率将变成 1470：1。也就是说在一个帧速度为 30 的项目里，每过 1470 个取样，才有一个会产生声音的变化。

上面的例子强调在项目中需要注意元件的采样率，以及按需求使用 Resample CHOP。除此之外，也有一些情况需要让输入输出数据使用不同的采样率。

4.5 时间片段化处理机制-Time Slicing

时间片段化是 TD 里面一个很独特的处理机制，我必须说，这很难一下理解明白。

一个片段单位是最后渲染帧和当前渲染帧之间的时段。把时间片段想象为一个动态的值，如果一个项目稳定运行在 60 帧，那么时间片段就是 1（因为并没有掉帧）。如果这个项目的实时渲染跟不上，每次渲染中间都丢 10 帧，那时间片段就是 10（当前渲染帧和上一个渲染帧之前实际上差了 10 帧）。

当存在掉帧现象时，时间片段化处理机制的存在就是为了平滑输出的 CHOP 数据。简单地说，时间片段化处理机制会在 CHOP 渲染的时候去考虑每个事件片段的 长度。这种机制可以看作一种自适应处理，当时间片的长度增加，CHOP 会根据丢失帧的数量做出补偿，调节帧数量，来实现平滑的输出。形成鲜明对比的是没有做时间片化处理的 CHOP。他们在最后一帧处理完数据后，不管中间丢了多少帧，都会直接跳到下一帧的数据 。只有 CHOP 元件能使用时间片段化处理机制。

在上面的例子里，如果时间轴以 30FPS 的速度持续运行，每个时间片段就是 1 帧的长度。如果在一秒钟内或者说 30 帧内，有两条线路的数值都是从 0 到 1，输出都是平滑型线路。但由于某些原因，每 10 帧才有一帧能被处理，两条线路的结果就会大不相同。在没有时间片化的线路中，在处理的那一帧，由于中间帧的丢失，数据会明显跳跃。 有做时间片化的 CHOP 会注意到每过 10 帧才有一帧能被处理，所以它会在当前帧和最后一帧之间插入中间帧。这样就可以保证不论发生什么，数据都是平滑的。

下图可以说明上面的例子。红线是时间片化后的输出，蓝线是没做片段化机制输出的帧，绿色垂直的线是渲染过的帧。